01

Token de cadastro e configuração de envio de email

Transcrição

[00:00] Na aula anterior vimos que mesmo os formulários de modelo, também armazenam outros dados, outros campos e aprendemos a recuperar e usar esses dados, como por exemplo na confirmação de senha, que é um campo que não existe no nosso modelo de usuário, mas que conseguimos pegar como parâmetro da requisição e utilizar para garantir que a senha e que a confirmação de senha são idênticas.

[00:25] Vimos também como usar o criptografador que vem embutido no play para garantir a segurança da senha do usuário, utilizando o algoritmo SHA-1 para encriptar a senha do usuário, só que ainda é muito fácil abusar do nosso sistema, porque podemos criar infinitos usuários. Digamos que eu crie o usuário com email marco@caelum.com.br e eu vou lá e eu crio também o usuário com email guilherme@caelum.com.br e eu tomo poder do usuário do Guilherme.

[01:00] Isso acontece porque não fizemos nenhum tipo de confirmação de que aquele email pertence a pessoa que está criando a conta, então eu estava pensando em criar aquela clássica confirmação por email que envia um link para o usuário por email para ele clicar e confirmar que aquele email pertence realmente a ele.

[01:17] Essa confirmação em geral inclui um token de cadastro e um marcador no usuário para indicar que o usuário foi verificado, então vamos fazer essas duas coisas. Primeiro eu criei aqui uma boolean verificada no nosso usuário, eu criei também já o getter e o setter dela, para podermos manipular ela quando quisermos.

[01:40] Agora precisamos criar também o modelo do nosso token de cadastro, então vamos lá, vou criar uma nova classe no pacote de modelos que se chama TokenDeCadastro e ela tem que estender a classe model do Ebean, modelo está criado, vamos configurar ela como entidade do banco, temos que dar um ID, temos que falar que o ID é gerado automaticamente.

[02:15] E o que mais vai ter o nosso token de cadastro? Ele tem que ter aquele código que mandamos para o usuário, então private String codigo, e ele tem que ter também uma ligação com usuário para indicar que o código está atrelado a um usuário e que aquele usuário tem um código, então private Usuario usuario. E como é uma relação única, um único usuário pode ter um código e vice-versa anotamos essa relação como @OneToOne.

[02:45] Em geral geramos getters e setters para todos os nosso campos, mas como não queremos que isso seja mutável, eu vou gerar somente os getters e setamos tudo dentro do construtor da classe, então vamos lá, vamos criar o construtor, public TokenDeCadastro que vai gerar tudo que precisamos aqui. Precisamos de um usuário, então vamos receber o usuário dentro do construtor, Usuario usuario e jogamos isso dentro do campo de usuário.

[03:21] E agora precisamos do código que é a única outra coisa que configuramos aqui, então this.codigo vai ser igual a o que? Pegamos algumas informações do usuário, adicionamos uma string aleatória e criptografamos tudo isso e garantimos a segurança dos dados do usuário, mas ainda tem alguma coisa que garante que vai ser única para ele.

[03:47] Então vamos fazer assim, vamos encriptar usando sha-1, vamos encriptar o que? usuario.getNome e podemos somar isso com usuario.getEmail, então agora já temos uma string longa o suficiente e única o suficiente para um dado usuário e adicionamos uma chave aleatória, um cookie aleatório, uma string que não temos controle sobre ela vai ser algo randômico, e criptografamos tudo isso e guardamos no código.

[04:23] Parece tudo ok, falta uma coisa, lembra que desabilitamos as evoluções do play? Então, não temos mais a geração do sql automático, então vamos precisar fazer o sql sozinho, eu vou copiar aqui o arquivo 3, gerar um 4, só que

eu não vou gerar o sql na mão, eu já fiz isso antes porque seria muito chato ficar passando por cada uma das coisas, eu vou pegar aqui da minha colinha.

[04:52] O que temos aqui? Temos o create table do nosso token de cadastro, que tem um ID, uma relação com o usuário e uma string código. Tem também aqui uma alteração da tabela do usuário com uma coluna verificado que é uma booleana e temos os drops das duas mudanças que acabamos de fazer, nosso sql parece que está certo, podemos vir aqui atualizar o nosso browser e ele vai pedir para fazer as evoluções.

[05:27] Depois que as evoluções forem concluídas o que mais que precisamos fazer? Precisamos efetivamente salvar o nosso token, gerar um token e salvar no banco, então vamos lá no Eclipse, vamos vir aqui logo depois de salvar o usuário no método salvaNovoUsuario, e geramos um token a partir desse usuário. New TokenDeCadastro, passando o usuário que acabamos de salvar, guarda isso numa variável, vou chamar essa variável de token e eu vou dar um token.save para guardar isso no banco.

[06:06] Então a partir daqui já temos um token salvo para esse usuário que acabamos de cadastrar, o que precisamos fazer agora? Precisamos enviar um email para o usuário, para enviar o email para o usuário vamos precisar fazer várias configurações.

[06:22] Primeira delas é baixar uma nova dependência, o enviador de e-mails do play, ou play mailer, então vamos vir aqui no build.sbt, vamos vir na parte de dependências e adicionamos a dependência do play mailer, que é essa dependência que eu vou disponibilizar para vocês, tanto na transcrição, quanto nos exercícios.

[06:48] Adicionamos aqui a dependência, podemos esquecer do build.sbt. Agora ainda temos mais uma configuração para fazer, toda vez que você quer enviar o email você tem que fazer a autenticação em um servidor de e-mails, então precisamos fazer toda essa configuração, essa configuração vai aonde? Na configuração do aplicativo, vamos vir até aqui no fundo e eu vou colar para vocês as configurações que temos aqui do enviador de e-mails do play.

[07:17] Essas configurações todas são as configurações do servidor e como ele vai reagir, mas isso não importa tanto para gente no desenvolvimento, porque vamos pegar e falar que toda essa configuração é basicamente um mock-up, é uma configuração falsa, essa configuração falsa faz o que? Em vez de você enviar o email de verdade usando seu servidor, o play vai identificar essa configuração e vai imprimir os dados do email como remetente, assunto e destinatário e o corpo do email, no registro dele, no log do console.

[07:53] Sem essa configuração você consegue efetivamente enviar e-mails, mas não é o que queremos aqui no nosso projeto em desenvolvimento, então é isso, se você quiser efetivamente enviar os e-mails, você remove essa configuração ou muda aqui para no, mas vamos deixar aqui yes.

[08:15] Depois de tudo configurado, vamos vir aqui matar nosso servidor e atualizar as configurações dele com um reload e quando ele terminar o comando eclipse, lembrando que isso serve para atualizar o class path do eclipse e adicionar as novas dependências nele. Agora que ele terminou eu subo o projeto de novo e limpo aqui o histórico.

[08:40] Agora eu posso atualizar o nosso projeto no eclipse e ele vai reconhecer as novas dependências. Podemos efetivamente mandar o nosso email, para mandar um email precisamos de um cliente de enviador de e-mails, no caso, mailer client, podemos, porque o play permite isso receber ele direto injetado aqui no nosso controler, então “private MailerClient enviador” do pacote “play.libs.mailer”.

[09:10] E agora para enviar um email pegamos esse enviador e enviamos um email, no caso vamos usar as palavras em inglês, o enviador vai chamar o método envia em inglês send, e um novo email do pacote “libs.mail”.

[09:33] Geramos um token para o usuário, configuramos o servidor de e-mails, salvamos o token no banco atrelado a esse usuário e enviamos um email, claro que esse email está cru, ele está super vazio, não configuramos nada nele,

então vamos fazer isso na aula que vem.